DC Motor Speed: Filtering and Control

1. Introduction

This Virtual Laboratory involves exercises using MATLAB and Simulink to perform modelling and simulation of dynamic systems representing a permanent magnet DC motor. Subsequently, analog and digital filters are introduced to reduce the effect of sensor noise affecting the motor speed.

You will follow a series of steps that will guide you to build relevant models and analyse system performance.

The submission of this coursework will be done via Blackboard. You will be asked to answer various questions and in some cases to upload the graphical output you generated.

Matlab coding procedure

- The most convenient way is to write code as .m format script file and execute it.
- Alternatively you may find it more convenient to use .mlx file format (*live script*) which displays the output in the same window alongside the relevant code
- In both cases the file can be split into sections (*insert section break*).
- Each section can be run separately. Beware that once set, variable values will remain unchanged until the code to reassign values is executed

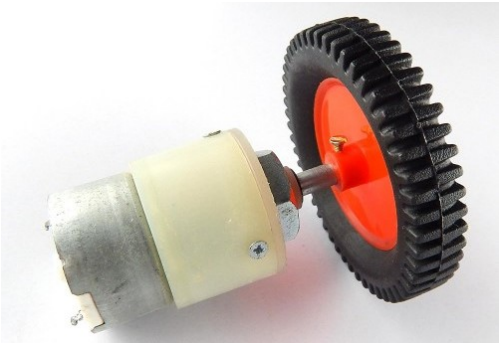
Steps to upload graphical output:

- Please upload all graphics on Blackboard as JPG image files (*.jpg)
- MATLAB figures can be saved as MATLAB figure files (*.fig) and converted to *.jpg using "`Save As...`" command.
- Alternatively you can use ***Printscreen*** or equivalent command to save a selected window or a portion of your computer screen.
    - On Windows you can use key combination <Shift+Win+s> to invoke the *Snip & Sketch* tool, and select the relevant portion of the screen.
    - Save the image a JPG file, and then
    - Upload the file on Blackboard as required

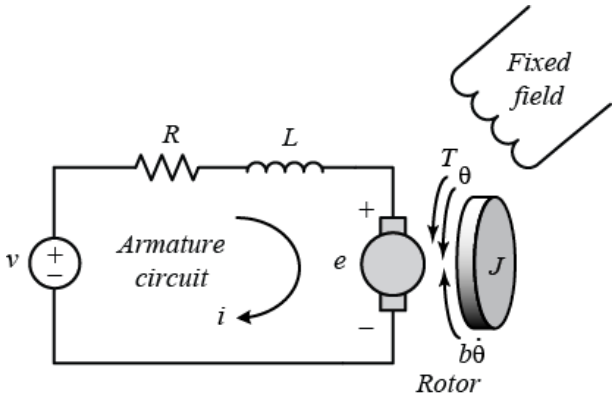
The exercises are based on those available on http://ctms.engin.umich.edu/CTMS/ but the model parameters have been changed, so your results will not be the same as those.

2. DC motor system

A common actuator in control systems and robotics is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion.



The electric circuit of the armature and the free-body diagram of the rotor are shown in the following figure.



We will assume that the input of the system is the voltage source (V) applied to the motor's armature. The system output is the rotational speed of the shaft $\dot{\theta}$, which we assume to be measured with a sensor (e.g. Hall effect sensor). **Sensor measurements are often affected by noise**, which can have electrical or mechanical causes. The rotor and shaft are assumed to be rigid. We further assume a viscous friction model due to the bearings and the wheel, that is, the friction torque is proportional to shaft angular velocity.

The physical parameters for our example are:

J	moment of inertia of rotor and wheel	0.1 kg.m^2
b	viscous friction constant of motor and wheel	0.2 N.m.s
K_e	electromotive force constant	0.02 V/rad/sec
K_t	motor torque constant	0.02 N.m/Amp
R	electric resistance	2 Ohm
L	electric inductance	0.4 H

In general, the torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torque is proportional to only the armature current i by a constant factor K_t as shown in the equation below. This is referred to as an armature-controlled motor.

(1) $T = K_t\, i$

The back emf, e, is proportional to the angular velocity of the shaft by a constant factor K_e.

(2) $e = K_e \dot{\theta}$

In SI units, the motor torque and back emf constants are equal, that is, $K_i = K_e$; therefore, we will use K to represent both the motor torque constant and the back emf constant.

3. Simulink model

This system will be modelled by summing the torques acting on the rotor inertia and integrating the acceleration to give velocity. Also, Kirchoff's laws will be applied to the armature circuit. First, we will model the integrals of the rotational acceleration and of the rate of change of the armature current.

(3) $$\int \frac{d^2\theta}{dt} dt = \frac{d\theta}{dt}$$

(4) $$\int \frac{d\,i}{dt} dt = i$$

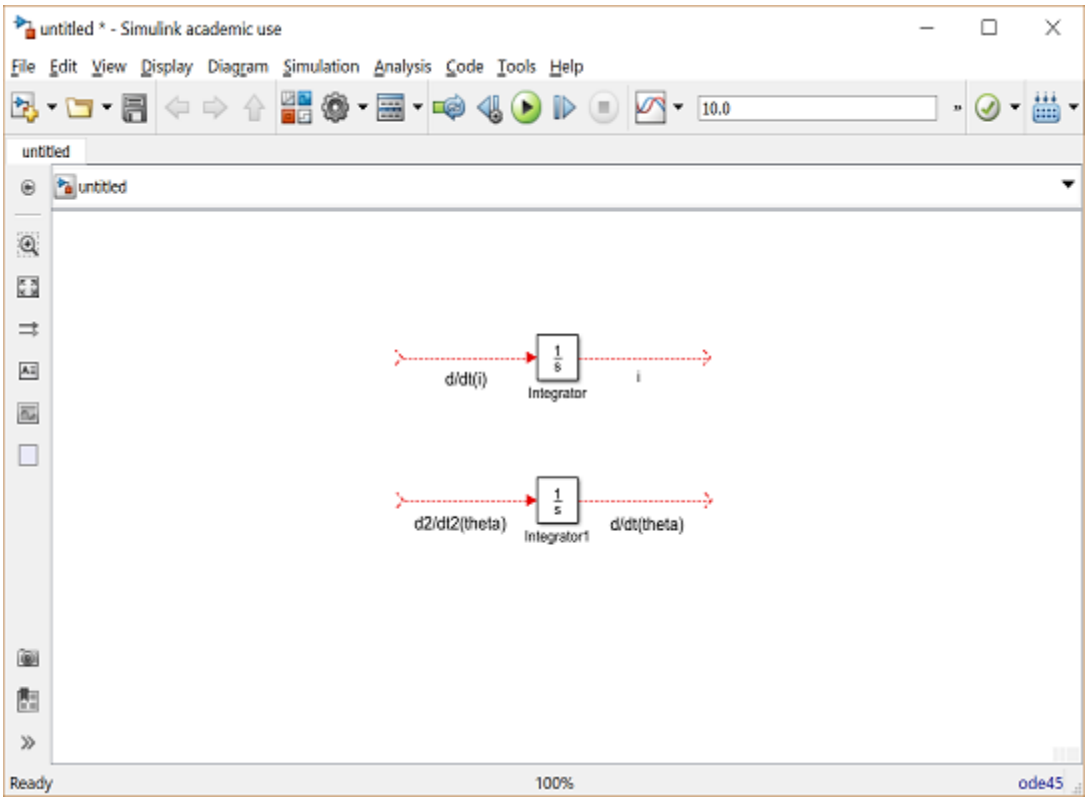
To build the simulation model, open Simulink and open a new model window. Then follow the steps listed below.

Insert an Integrator block from the Simulink/Continuous library and draw lines to and from its input and output terminals.

Label the input line "d2/dt2(theta)" and the output line "d/dt(theta)" as shown below. To add such a label, double-click in the empty space just below the line.

Insert another Integrator block above the previous one and draw lines to and from its input and output terminals.

Label the input line "d/dt(i)" and the output line "i".



Next, we will apply Newton's law and Kirchoff's law to the motor system to generate the following equations:

(5) $$J\frac{d^2\theta}{dt} = T - b\frac{d\theta}{dt} \rightarrow \frac{d^2\theta}{dt} = \frac{1}{J}\left(K_i\, i - b\frac{d\theta}{dt}\right)$$

(6) $$L\frac{di}{dt} = -Ri + V - e \rightarrow \frac{di}{dt} = \frac{1}{L}\left(-Ri + V - K_e\frac{d\theta}{dt}\right)$$

The angular acceleration is equal to $1/J$ multiplied by the sum of two terms (one positive, one negative). Similarly, the derivative of current is equal to $1/L$ multiplied by the sum of three terms (one positive, two negative). Continuing to model these equations in Simulink, follow the steps given below.

Insert two Gain blocks from the Simulink/Math Operations library, one attached to each of the integrators.

Edit the Gain block corresponding to angular acceleration by double-clicking it and changing its value to "1/J".

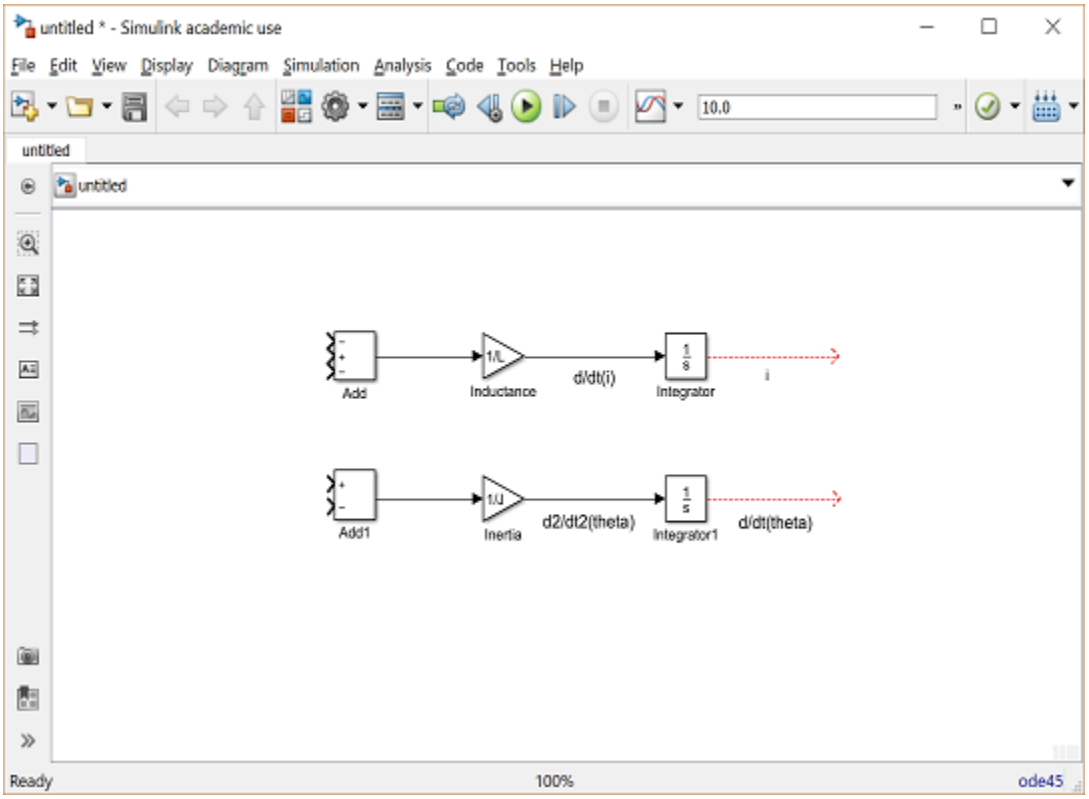
Change the label of this Gain block to "Inertia" by clicking on the word "Gain" underneath the block.

Similarly, edit the other Gain's value to "1/L" and its label to "Inductance".

Insert two Add blocks from the Simulink/Math Operations library, one attached by a line to each of the Gain blocks.

Edit the signs of the Add block corresponding to rotation to "+-" since one term is positive and one is negative.

Edit the signs of the other Add block to "-+-" to represent the signs of the terms in the electrical equation.



Now, we will add in the torques which are represented in the rotational equation. First, we will add in the damping torque.

Insert a Gain block below the "Inertia" block. Next right-click on the block and select Rotate & Flip > Flip Block from the resulting menu to flip the block from left to right. You can also flip a selected block by holding down Ctrl-I.

Set the Gain value to "b" and rename this block to "Damping".

Tap a line (hold Ctrl while drawing or right-click on the line) off the rotational Integrator's output and connect it to the input of the "Damping" block.

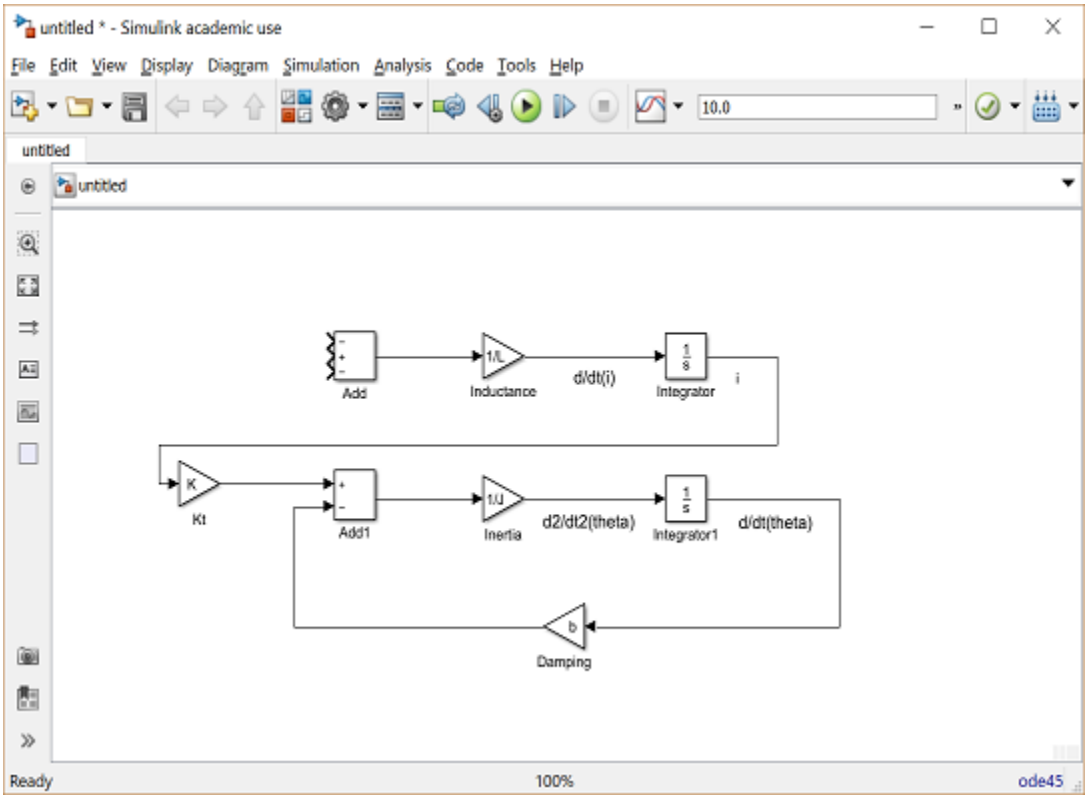
Draw a line from the "Damping" block output to the negative input of the rotational Add block.

Next, we will add in the torque from the armature.

Insert a Gain block attached to the positive input of the rotational Add a block with a line.

Edit its value to "K" to represent the motor constant and Label it "Kt".

Continue drawing the line leading from the current Integrator and connect it to the "Kt" block.



Now, add in the voltage terms which are represented in the electrical equation. First, add in the voltage drop across the armature resistance.

Insert a Gain block above the "Inductance" block and flip it from left to right.

Set the Gain value to "R" and rename this block to "Resistance".

Tap a line off the current Integrator's output and connect it to the input of the "Resistance" block.

Draw a line from the "Resistance" block's output to the upper negative input of the current equation Add block.

Next, we will add in the back emf from the motor.

Insert a Gain block attached to the other negative input of the current Add block with a line.

Edit it's value to "K" to represent the motor back emf constant and Label it "Ke".

Tap a line off the rotational Integrator's output and connect it to the "Ke" block.

Add **In1** and **Out1** blocks from the Simulink/Ports & Subsystems library and respectively label them "Voltage" and "Speed".

The final design should look like the example shown in the figure below.

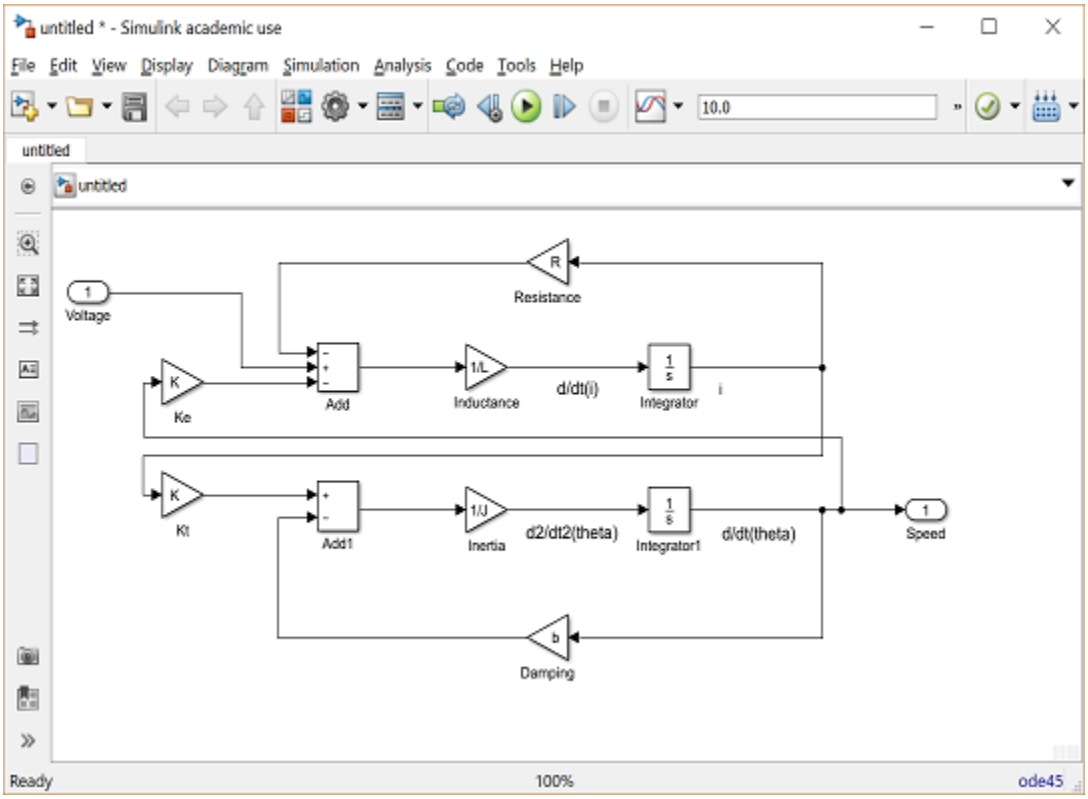
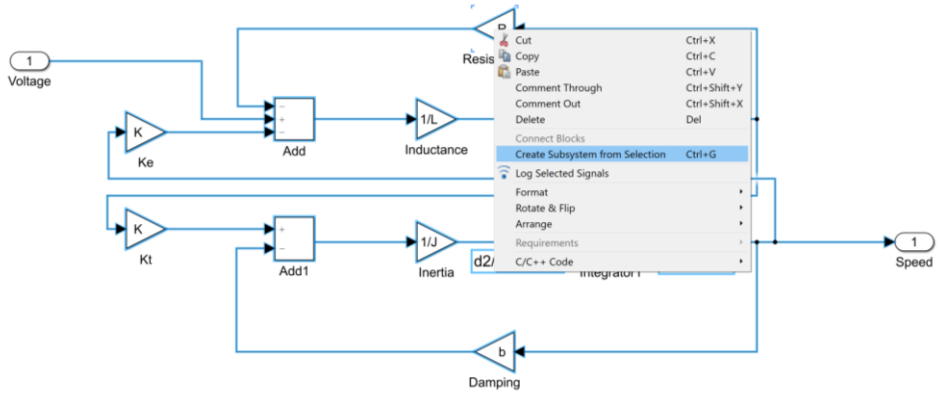
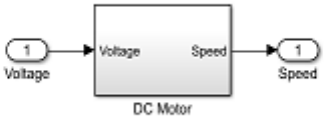


Figure 3.1: Simulink model of the DC motor

In order to save all of these components as a single subsystem block, first select all of the blocks, then select **Create Subsystem from Selection** after right-clicking on the selected portion. Name the subsystem "DC Motor" and then save the model, as indicated in the following figure.



The resulting block diagram should appear as follows.



3.1 Open-loop step response

The **open-loop** response of the DC motor (without any feedback control) to a step input voltage of 1 Volt is simulated in MATLAB.

Recall that the physical parameters have to be set if they have not previously been defined in the workspace.

b = 0.2;

J = 0.1;

K = 0.02;

L = 0.4;

R = 2;

Save your file as `DCMotor.slx` (select **Save As** from the **File** menu). MATLAB will extract the linear model from the saved model file, not from the open model window. At the MATLAB prompt, enter the following commands to compute the transfer function of the DC motor:

```
b = 0.2;
J = 0.1;
K = 0.02;
L = 0.4;
R = 2;

u = 1;
[num,den] = linmod('DCMotor');
P=tf(num,den);
```

To verify the model extraction, we will generate an open-loop step response in MATLAB. We will simulate a step input of 1 V and we will generate the Bode plot. Enter the following command in MATLAB.

```
figure(1);
step(u*P);

figure(2);
bode(P);
```

You should obtain the following step response plot and the Bode plot shown below

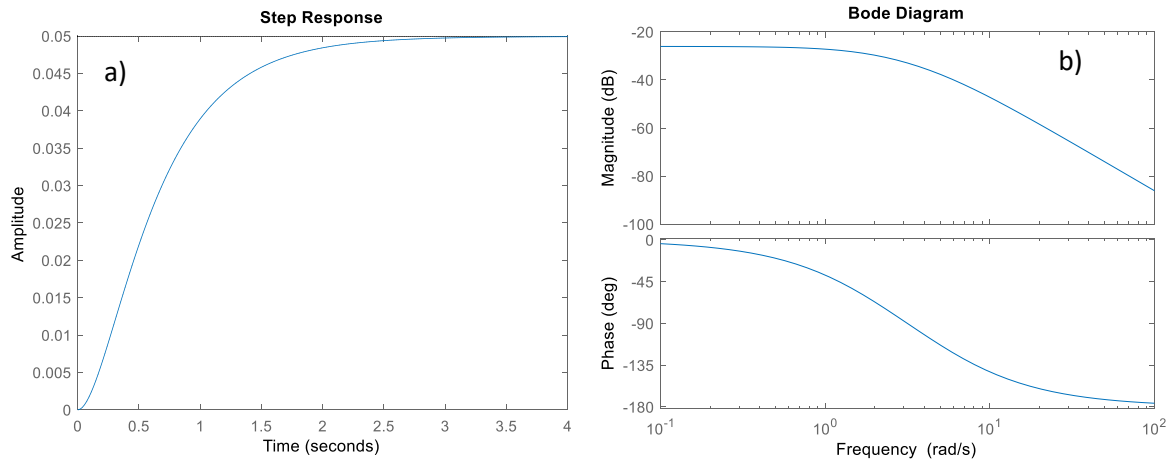
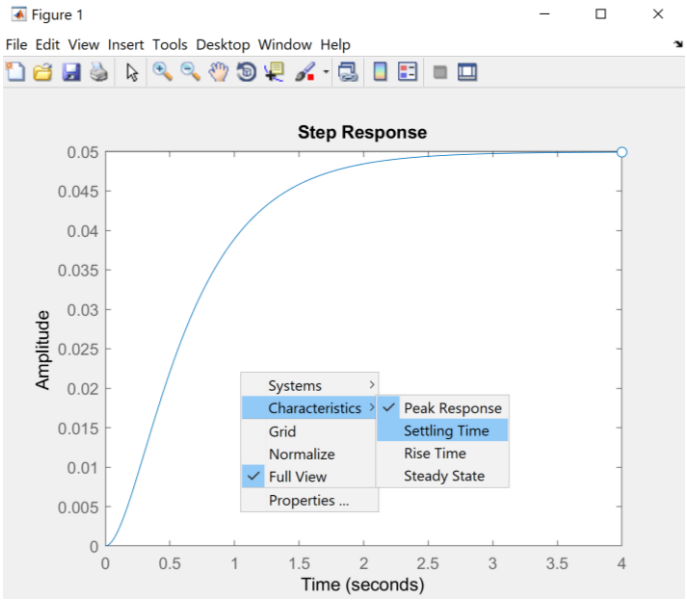


Figure 3.2: Step response of the DC motor (a); Bode plot (b)

At this point we can assess the performance of our DC motor by using some well-known performance indicators:

- Rise time
- Settling time
- Steady state

To measure these values, **right click** in the plot of the step response and select **Characteristics**, and choose the corresponding option from the drop-down list, as indicated in the following figure.



Questions:

Q 1 Upload the DC motor model as in Fig. 3.1 (`DCMotor.slx`) on **Blackboard**

Q 2 Upload the step response plot (see Fig. 3.2.a) on **Blackboard**

Q 3 Upload the Bode plot (see Fig. 3.2.b) on **Blackboard**

For the DC motor system, identify the following quantities:

Q 4(a) Steady state speed

Q 4(b) Rise time

Q 4(c) Settling time

4. Analog filter design

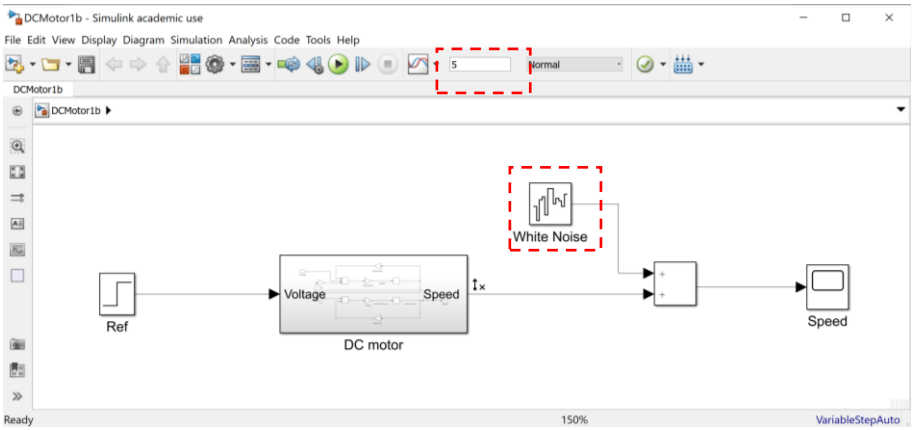
In the next step we shall simulate the effect of measurement noise and we shall assess the effect of filtering.

Insert a block **Add** from the library **Simulink \ Math Operations**. Double click on it and enter the values "++" in the **List of Signs**. This will create two positive input ports. Connect one of the input ports to the Speed.

Insert a block **Band-limited White noise** from the library **Simulink \ Sources** and connect it to the remaining input of the Add block. Double click on the block and set the values Noise Power to [0.0000001], Sample time to 0.001, and Seed to 1 (these values are used here for illustrative purposes). Insert a block **Scope** from the library **Simulink \ Sinks** and connect it to the line Speed. Set the number of channels to 1.

Insert a block **Step** from the library **Simulink \ Sources** and connect it to the positive input of the block **Add.** Double click on the block and set the values **Step time** to 0, **Initial value** to 0, **Final value** to 1.

Connect the negative input of the Add block to the line Speed. Save your file as `DCMotor2.slx` (select **Save As** from the **File** menu). The resulting block diagram is shown in the figure below.



Run the simulation setting the duration to **5 seconds** and then double click on the Scope to see the result, which should be similar to the following Figure 4.1.

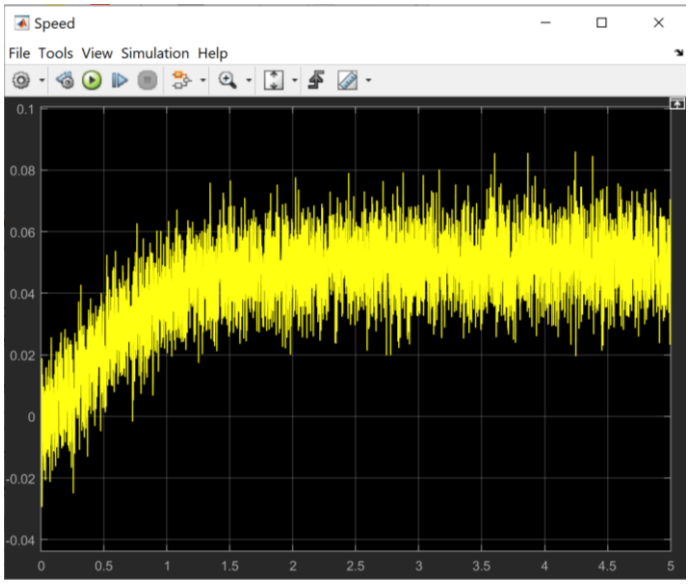


Figure 4.1: Step response of the Simulink model of the DC motor with white noise.

It is clear from Figure 4.1 that the white noise has a negative effect on the measured speed, thus filtering becomes necessary.

4.1 Low-pass analog filters

The prototype low-pass filter is based upon the magnitude-squared of the frequency response function $|H(j\Omega)|^2$, or the frequency response power function. The phase response of the filter is not considered here. We begin by defining tolerance regions on the power frequency response plot, as shown in Figure 4.2.

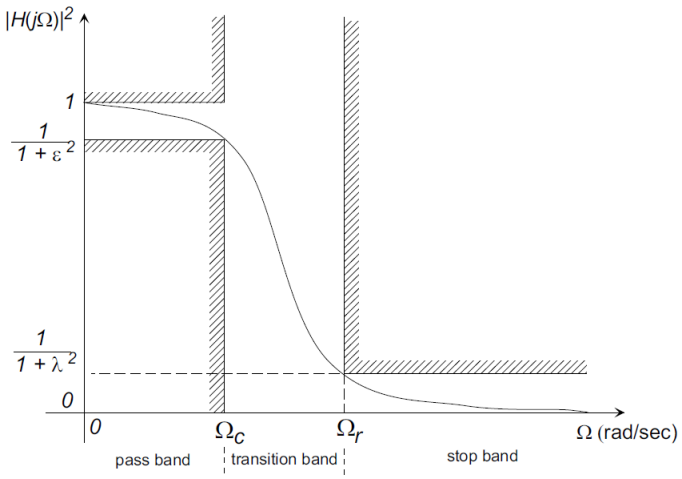


Figure 4.2: Tolerance regions in the frequency response plot.

The filter specifications are that

(7) $1 \geq |H(j\Omega)|^2 > \frac{1}{1+\epsilon^2}$ for $|\Omega| < \Omega_c$, corresponding to $\alpha_P = 20\log\left(\frac{A_0}{A_1}\right)$, with $A_1 = H(j\Omega_c)$, in the lecture

(8) $|H(j\Omega)|^2 < \frac{1}{1+\lambda^2}$ for $|\Omega| > \Omega_r$, corresponding to $\alpha_S = 20\log\left(\frac{A_0}{A_2}\right)$, with $A_2 = H(j\Omega_r)$, in the lecture

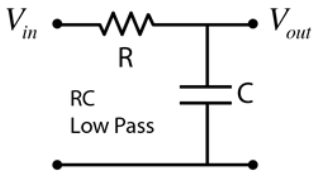
where Ω_c is the cut-off frequency (or passband frequency), Ω_r is the rejection frequency (or stopband frequency), and ϵ and λ are design parameters that select the filter attenuation at the two critical frequencies. This notation is alternative to that used in the lecture notes (i.e. α_P and α_S) and it is introduced here since both are common in engineering practice. For example, if $\epsilon = 1$, at Ω_c the power response is $|H(j\Omega)|^2 = 0.5$, the -3 dB response frequency. In general we expect the response function to be monotonically decreasing in the transition band.

The filter functions examined here will be of the form

(9) $|H(j\Omega)|^2 < \frac{1}{1+f^2(\Omega)}$,

where $f(\Omega) \to 0$ as $\Omega \to 0$, and $f(\Omega) \to \infty$ as $\Omega \to \infty$, to generate a low-pass filter action.

Analog filters can be implemented with resistors and capacitors and, in case of active filters, with operational amplifiers. The figure below shows a first-order low-pass filter. High order filters can be implemented by connecting multiple 1st order filters in series.



$$|V_{out}| = |V_{in}| \frac{1}{\sqrt{1+\omega^2 R^2 C^2}}$$

To start with, we consider a **first order low-pass filter** with transfer function

(10) $$F(s) = \frac{1}{s/\Omega_c + 1}$$

The corner frequency Ω_c is expressed in rad / s and represents the frequency values at which the filters gives a –3 dB attenuation. For illustrative purposes we choose a tentative value $\Omega_c = 50$ rad / s. Since the prescribed speed of the DC motor is 1 rad/s, this ensures that the system dynamics is not filtered out of the Speed signal and reduces the effect of the noise on the measurement.

Additional details on low-pass filters can be found in https://web.stanford.edu/~boyd/ee102/conv_demo.pdf

To implement a filter in the block diagram, open the file `DCMotor2.slx`, insert a block **Transfer Fcn** from the library **Simulink \ Continuous** and connect its input to the output of the Add. Rename the block to **Filter** and double click on it to set the values Numerator to 1 and Denominator to [1/50 1]. This corresponds to a first-order low-pass filter with corner frequency 50 rad/s. Save the file as `DCMotor2a.slx`.

Delete the line between the Add block and the Scope and draw new lines as in Figure 4.3 below.

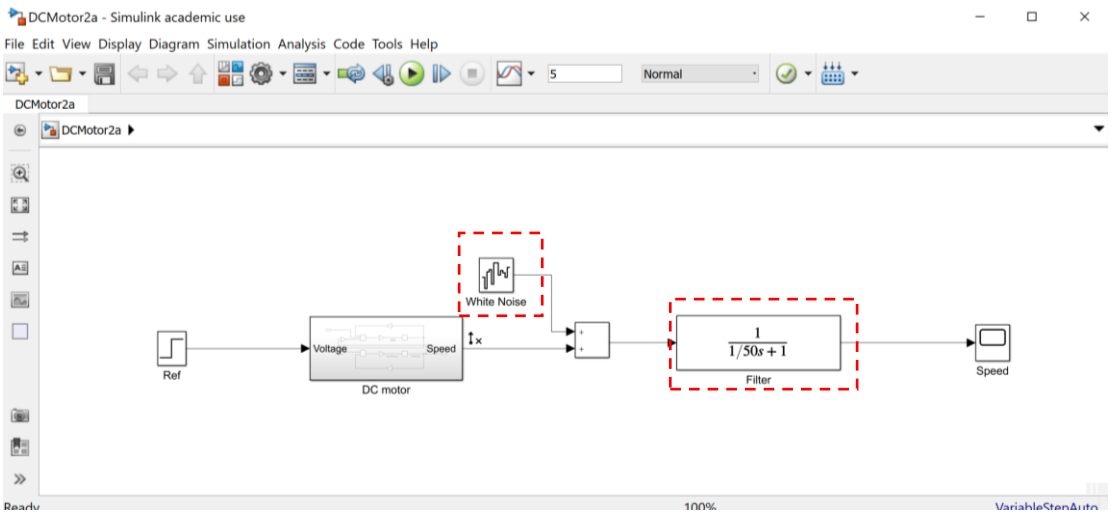


Figure 4.3: Simulink model of the DC motor with noise source and low-pass filter.

Save your file, run the simulation again, and the double click on the Scope to see the result, which should be similar to the following Figure 4.4.

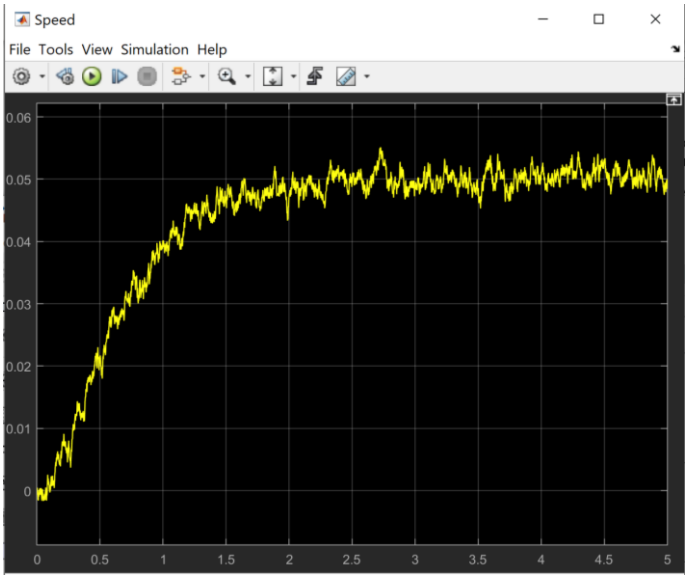


Figure 4.4: Step response of the Simulink model of the DC motor with white noise source and low-pass filter.

Introducing the low-pass filter attenuates the measurement noise affecting, as shown in Figure 4.4. However, the filter also affects the dynamics of the system by introducing a phase lag. To highlight this effect, modify the Simulink file `DCMotor.slx` by adding the low-pass filter, as shown in the figure below, and save it as `DCMotor1a.slx`.

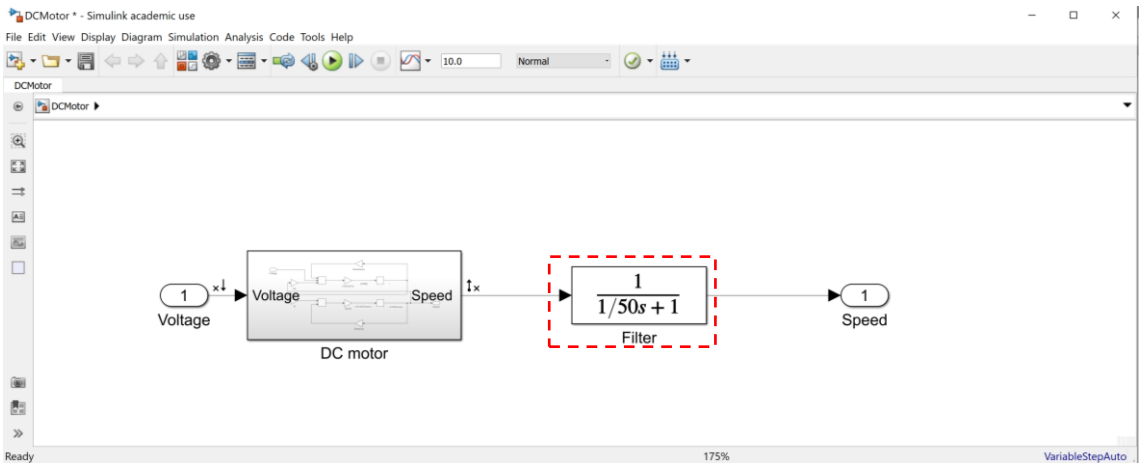


Figure 4.5: Simulink model of the DC motor with low-pass filter.

At the MATLAB prompt, enter the following commands to compute the transfer function of the DC motor with the low-pass filter. To assess the effect of the filter on the system dynamics we will simulate a step input of 1 V and we will generate the Bode plot.

```
[num,den] = linmod('DCMotor1a');
P=tf(num,den);

figure(1);
step(u*P);

figure(2);
bode(P)
```

You should obtain the step response plot and the Bode plot shown below. Note that the phase has changed and its value at high frequency is close to -270 deg. This is as excepted since the DC motor is 2nd order while the filter is 1st order.

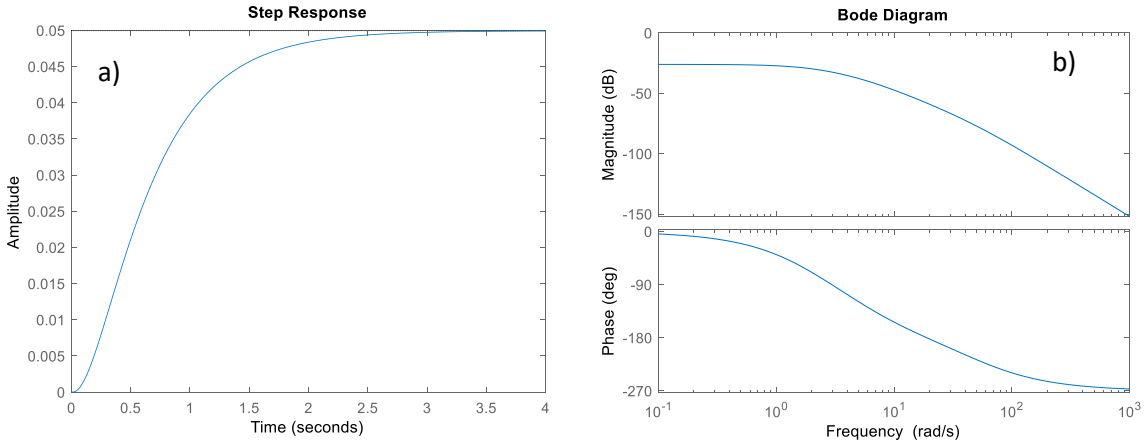
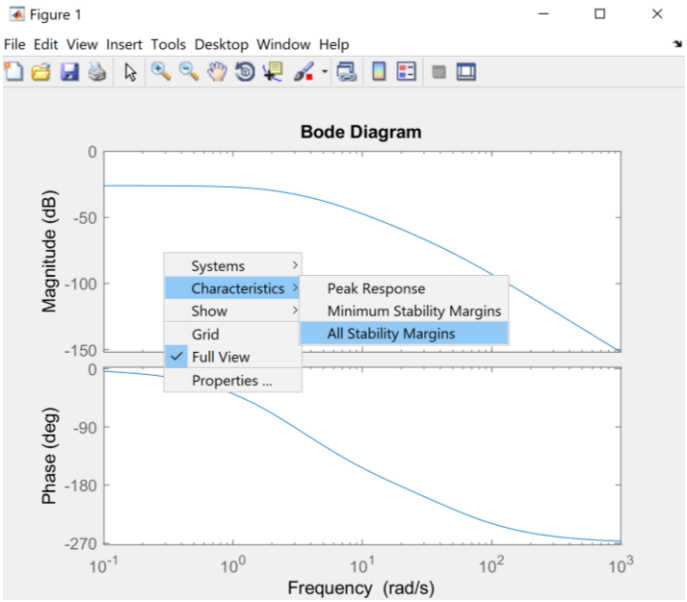


Figure 4.6: Step response of the DC motor with first-order low-pass filter (a); Bode plot (b)

Right click on the Bode diagram and select **Characteristics** and **All stability margins**, to see the gain margin and the phase margin.



Questions:

For the DC motor system with noise source and filter (file `DCMotor2a.slx`), upload the following files on **Blackboard**:

Q 5 Upload the step response plot as in Fig. 4.4 on **Blackboard**

Double click on the Filter block in `DCMotor2a.slx` and change the denominator to [1/30 1].

Q 6 Upload the new step response plot on **Blackboard**

Plot the Bode diagram for the DC motor system with first-order low-pass filter (file `DCMotor1a.slx`) shown in Figure 4.6.b:

Q 7 Is the system stable?

Q 8 What is the gain margin in dB?

Q 9 What is the phase cross-over frequency in rad/s ?

4.2 Butterworth filters

The most used filters include the Butterworth and the Chebyshev filter. Other types of filters are the Bessel and the elliptic filters, which however are less used.

The Butterworth filter is defined by the power gain

(11) $$|H(j\Omega)|^2 = \frac{1}{1+\epsilon^2(\Omega/\Omega_c)^{2N}},$$

where N is a positive integer defining the filter order. Although the notation is different, equation 11 has a similar structure to Equation 10 in Chapter 3 of the lecture notes. Note that λ does not appear in this formulation, but clearly N and λ are interrelated, since at $\Omega = \Omega_r$ we have

(12) $$\frac{1}{1+\lambda^2} \geq \frac{1}{1+\epsilon^2(\Omega_r/\Omega_c)^{2N}},$$

which may be solved to show

(13) $$N \geq \frac{\log(\lambda/\epsilon)}{\log(\Omega_r/\Omega_c)}.$$

The power response function of Butterworth filters for N =1...5 is shown in Figure 4.7. The poles of a Butterworth filter are located on a semicircle around the origin of the complex plane.

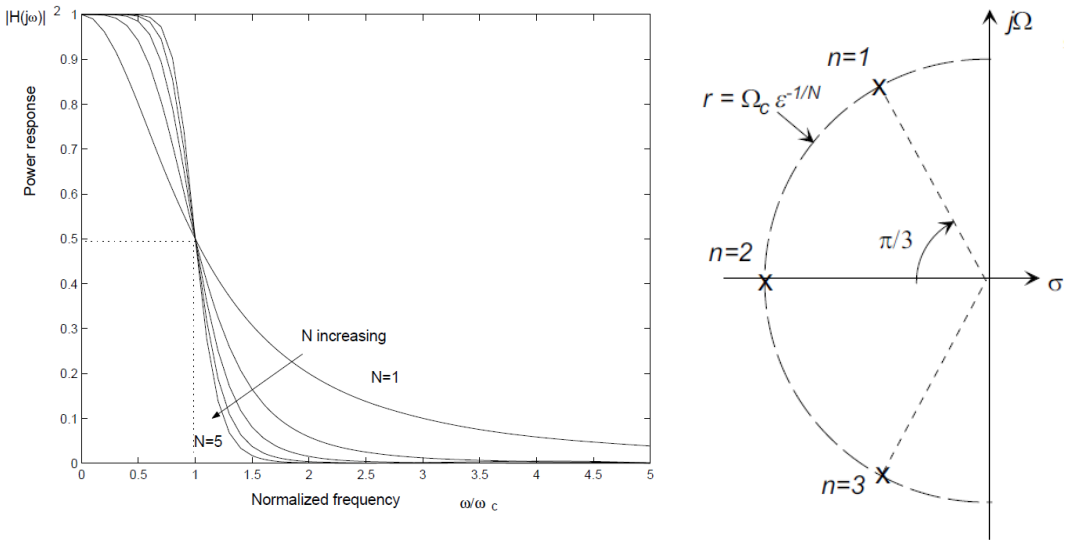


Figure 4.7: Power response of Butterworth filters for N =1...5 (ϵ = 1), and poles of a third-order (N = 3) Butterworth filter.

We shall now design a Butterworth filter to fulfil a set of requirements. With respect to Figure 4.2, we set the **cut-off frequency to 50 rad/s**, the **attenuation at cut-off frequency to 3 dB**, the **stopband frequency to 70 rad/s**, and the **attenuation at stopband frequency to 6 dB**.

In particular, the stopband frequency and the corresponding attenuation are additional to the first-order filter requirements seen in equation (9) and are necessary to define the filter order.

Add the following instructions to your m-file:

```
%% Butterworth filter design:
% wc -passband cut-off frequency (rad/s)
% Rc -attenuation at passband cut-off (dB) (Rc > 0)
% ws -stopband frequency (rad/s)
% Rs -attenuation at stopband (dB) (Rs > 0)

wc=50;
Rc=3;
ws=70;
Rs=6;

epsilon = sqrt(10^(Rc/10)-1);
lambda = sqrt(10^(Rs/10)-1);
N_Bw = ceil(log(lambda/epsilon)/log(ws/wc));
disp(N_Bw);

[numB,denB] = butter(N_Bw,wc,'s');
H = tf(numB,denB);
p_Bw=pole(H);
bode(H);          % Bode plot
disp(p_Bw);
```

To implement the above filter in the block diagram, double click on the block **Transfer Fcn** in the Simulink models `DCMotor1a.slx` and `DCMotor2a.slx` and set the values Numerator to `numB` and the Denominator to `denB` (this is the same as typing the values provided by the command).

```
[numB,denB] = butter(N_Bw,wc,'s')
```

Save your files with a new name (`DCMotor1b.slx` and `DCMotor2b.slx`).
Run the simulation in `DCMotor2b.slx`, and then double click on the Scope to see the result, which should be similar to the following Figure 4.8. Compared to Figure 4.4, the Butterworth filter has reduced the high-frequency noise to a larger extent. Some low-frequency noise remains, however it can be further reduced by employing a smaller cut-off frequency.

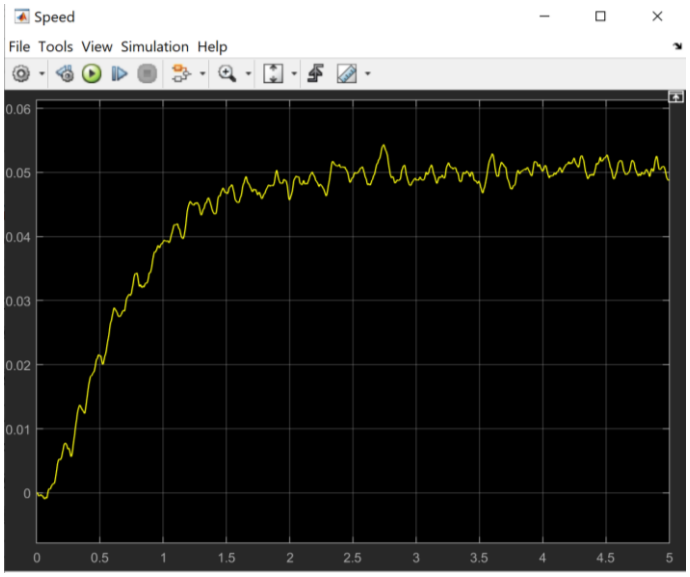


Figure 4.8: Step response of Simulink model with Butterworth filter.

To assess the effect of the filter on the system dynamics we will simulate a step input of 1 V and we will generate the Bode plot from the model `DCMotor1b.slx`.

```
[num,den] = linmod('DCMotor1b');
P=tf(num,den);

figure(1);
step(u*P);

figure(2);
bode(P)
```

You should obtain the step response plot and the Bode plot shown below. Note that the phase plot as changed and its value at high frequency is close to -360 deg.

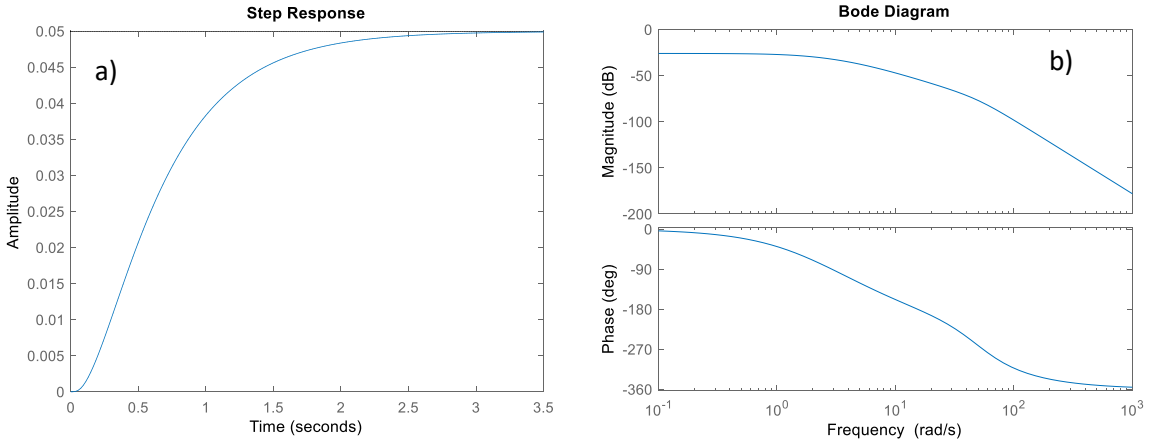


Figure 4.9: Step response of the DC motor with Butterworth filter (a); Bode plot (b)

Questions:

Q 10 What is the filter order `N_Bw` that meets the requirements?

`N_Bw = 1`

`N_Bw = 2`

`N_Bw = 3`

`N_Bw = 4`

Q 11 What is the location of the poles of the filter's transfer function?

s = -2.4323 and s = 0

s = -35.3553 ±35.3553i

s = 0.0000 ± 59.4957i and s = 0

s = -35.3553 ±35.3553i and s = 0

Q 12 Upload the bode plot of the Butterworth filter to **Blackboard**

Estimate the following values from the Bode diagram for the DC motor system with Butterworth filter (`DCMotor1b.slx`) as shown in Figure 4.9.b:

Q 13 Is the filter stable?

Q 14 What is the gain margin in dB?

Q 15 What is the phase cross-over frequency in rad/s ?

Change the stopband frequency to `ws=60` in your m-file.

Q 16 What is the filter order `N_Bw` now?

`N_Bw = 1`

`N_Bw = 2`

`N_Bw = 3`

`N_Bw = 4`

4.3 Chebyshev filters

The order of a filter required to meet a low-pass specification may often be reduced by relaxing the requirement of a monotonically decreasing power gain with frequency, and allowing "ripple" to occur in either the pass-band or the stop-band. The Chebyshev filters allow these conditions:

(14) Type 1 $$|H(j\Omega)|^2 = \frac{1}{1+\epsilon^2 T_N^2(\Omega/\Omega_c)}$$

(15) Type 2 $$|H(j\Omega)|^2 = \frac{1}{1+\epsilon^2 (T_N^2(\Omega_r/\Omega_c)/T_N^2(\Omega_r/\Omega))}$$ (for information only, not part of the lab)

$T_N(x)$ is the Chebyshev polynomial of degree N. The form of the Type 1 power gain is similar to the Butterworth filter, where the function $T_N(\Omega/\Omega_c)$ has replaced $(\Omega/\Omega_c)^N$. The Type 1 filter produces an all-pole design with slightly different pole placement from the Butterworth filters, allowing resonant peaks in the passband to introduce ripple, while the Type 2 filter introduces a set of zeros on the imaginary axis above Ω_r, causing a ripple in the stop-band (Type 2 is not part of this course, but it is often found in engineering practice).

The Chebyshev polynomials are defined recursively as follows

(16) $T_0(x) = 1$

(17) $T_1(x) = x$ In the lecture we only consider this type of polynomial (first kind).

(18) $T_2(x) = 2x^2 - 1$

(19) $T_3(x) = 4x^3 - 3x$

(20) $T_N(x) = 2xT_{N-1}(x) - T_{N_2}(x), N > 1$

with alternate definitions $T_N(x) = \cos(N\cos^{-1}(x)) = \cosh(N\cosh^{-1}(x))$.

Chebyshev Type 1 design

The required order for a Type 1 Chebyshev filter is

(21) $N \geq \frac{\cosh^{-1}(\lambda/\epsilon)}{\cosh^{-1}(\Omega_r/\Omega_c)}$.

The poles are placed on an ellipse with major axis coincident to the imaginary axis, as show in the following Figure 4.10.

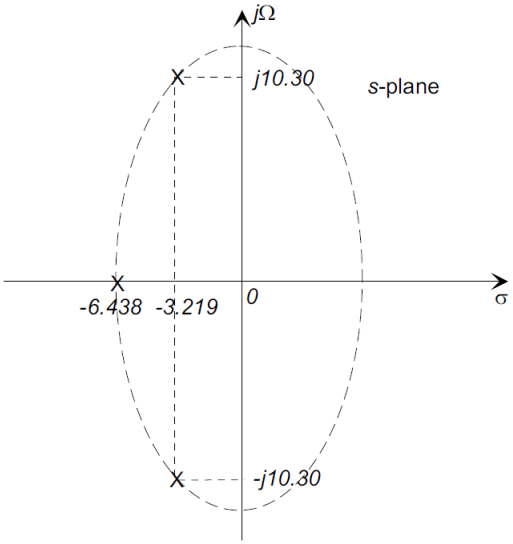


Figure 4.10: Poles of a third-order Type 1 Chebyshev filter.

We shall now design a Type 1 Chebyshev filter for the same requirements defined before, that is a **cut-off frequency of 50 rad/s**, the **attenuation at cut-off frequency is 3 dB**, the **stopband frequency is 70 rad/s**, and the **attenuation at stopband frequency is 6 dB**. We set the **peak-to-peak ripple** in the passband to $\boldsymbol{R_p = 3}$.

Add the following instructions to your m-file

```
%% Chebyshev Type 1 filter design:
% wc -passband cut-off frequency (rad/s)
% Rc -attenuation at passband cut-off (dB) (Rc > 0)
% ws -stopband frequency (rad/s)
% Rs -attenuation at stopband (dB) (Rs > 0).
% Rp -peak-to-peak ripple in the passband (dB) (Rp > 0).
close all;clc;

wc=50;
Rc=3;
ws=70;
Rs=6;
Rp=3;

epsilon = sqrt(10^(Rc/10)-1);
lambda = sqrt(10^(Rs/10)-1);
N_Cb1 = ceil(acosh(lambda/epsilon)/acosh(ws/wc));
disp(N_Cb1);

[numC1,denC1] =cheby1(N_Cb1,Rp,wc,'s');
H1 = tf(numC1,denC1);
p_Cb1=pole(H1);
bode(H1);         % Bode plot
disp(p_Cb1);
```

To implement the above filter in the block diagram, double click on the block **Transfer Fcn** in the Simulink models `DCMotor1b.slx` and `DCMotor2b.slx` and set the values Numerator as `numC1` and Denominator as `denC1`.

Save the files as `DCMotor1c.slx` and `DCMotor2c.slx`.

Run the simulation again, and the double click on the Scope to see the result, which should be similar to the following Figure 4.11.

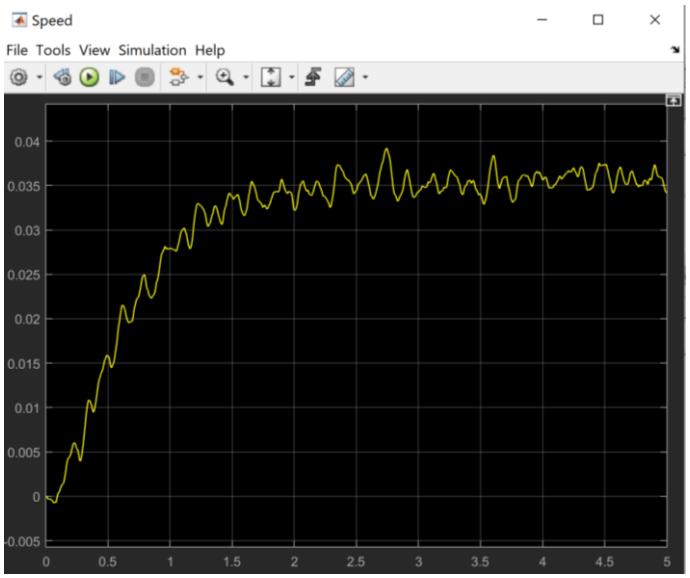


Figure 4.11: Step response of Simulink model with Chebyshev Type 1 filter.

To assess the effect of the filter on the system dynamics we will simulate a step input of 1 V and we will generate the Bode plot from the file `DCMotor1c.slx`.

```
[num,den] = linmod('DCMotor1c');
P=tf(num,den);

figure(1);
step(u*P);

figure(2);
bode(P)
```

You should obtain the following step response plot and the Bode plot shown below. Note that the phase plot is different from that of the Butterworth filter. In particular, the steady state is value is different due to the ripple in the Chebyshev filter.

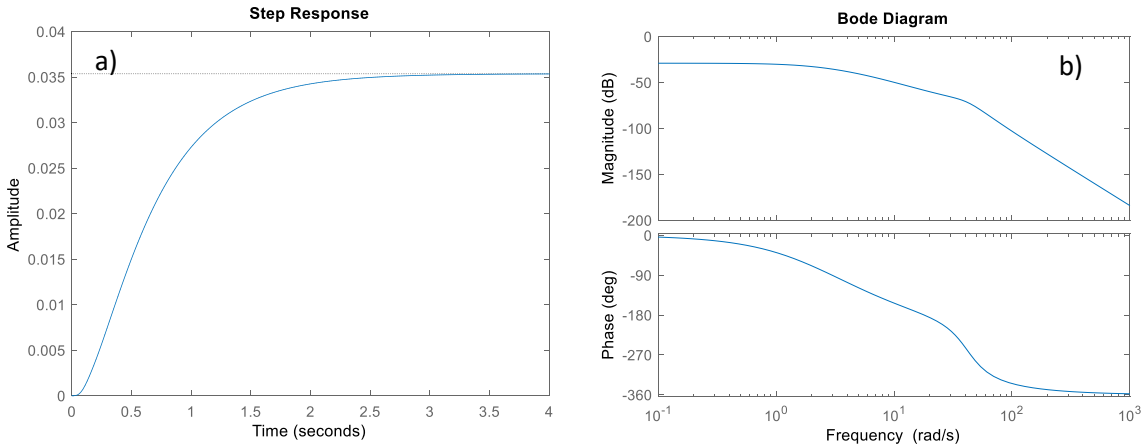


Figure 4.12: Step response of the DC motor with Chebyshev Type 1 filter (a); Bode plot (b)

Questions:

Q 17 What is the filter order `N_Cb1` that meets the requirements?

`N_Bw = 1`

`N_Bw = 2`

`N_Bw = 3`

`N_Bw = 4`

Q 18 What is the location of the poles of the filter's transfer function?

s = -35.3553 ±35.3553i

s = -16.1225 ±38.8579i

s = -16.1225 ±38.8579i and s = 0

s = -35.3553 ±35.3553i and s = -16.1225 ±38.8579i

Q 19 Upload the bode plot of the Chebyshev Type 1 filter to **Blackboard**

Estimate the following values from the Bode diagram of the DC motor system with Type 1 Chebyshev filter (`DCMotor1c.slx`) as shown in Figure 4.12:

Q 20 Is the system stable?

Q 21 What is the gain margin in dB?

Q 22 What is the phase cross-over frequency in rad/s ?

Change the stopband frequency to `ws=60` in your m-file.

Q 23 What is the filter order `N_Cb1` now?

`N_Bw = 1`

`N_Bw = 2`

`N_Bw = 3`

`N_Bw = 4`

4.4 Closed-loop control

In this section, we shall design a controller to ensure that the DC motor with the Butterworth filter meets a set of performance specifications. We choose the Butterworth filter for illustrative purposes, but the same procedure could be repeated for the other filters.

When the reference signal **r** is **1**, the motor should reach a velocity of **1 rad/s**. In this application, a 10% overshoot, a 2 s settling time, and 2% steady-state error on the velocity are sufficient.
Keeping the above in mind, we have proposed the following design criteria for this problem:

- Rise time < 1 s
- Overshoot < 10%
- Settling time < 1.5 s
- Steady state error < 2%
- Desired speed 1 rad / s

We will compute the poles of the transfer function corresponding to the DC motor model with the Butterworth filter from the file `DCMotor1b.slx`.

Double click on the block **Transfer Fcn** in the Simulink model `DCMotor1b.slx` and set the values Numerator to `numB` and the Denominator to `denB`. Save the file and repeat the MATLAB commands given below:

```
[num,den] = linmod('DCMotor1b');
P=tf(num,den);
zpk(P)
```

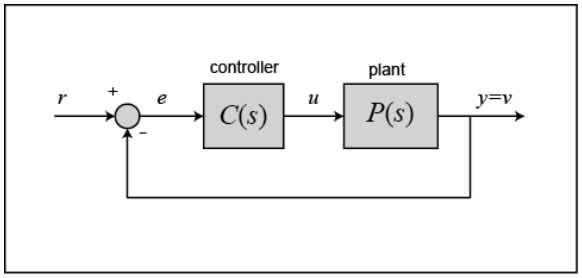
```
                  1250

  ----------------------------------------

  (s+4.997) (s+2.003) (s^2 + 70.71s + 2500)
```

Note that the plant is 4th order since it consists of the DC motor and of the Butterworth filter.

The block diagram of a typical unity feedback system is shown below.



As highlighted before, the transfer function of the plant is 4th order.

The transfer function of a PID controller is

(22) $$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

In general, we can define a PID controller using MATLAB's **pid controller object**, as shown in this example:

```
C = pid(Kp,Ki,Kd)
```

Let K_p equal 100, K_i equal 150, and K_d equal 10 and repeat the MATLAB commands given below.

```
Kp = 100;
Ki = 150;
Kd = 10;
C = pid(Kp,Ki,Kd);
T = feedback(C*P,1);
step(T);
```

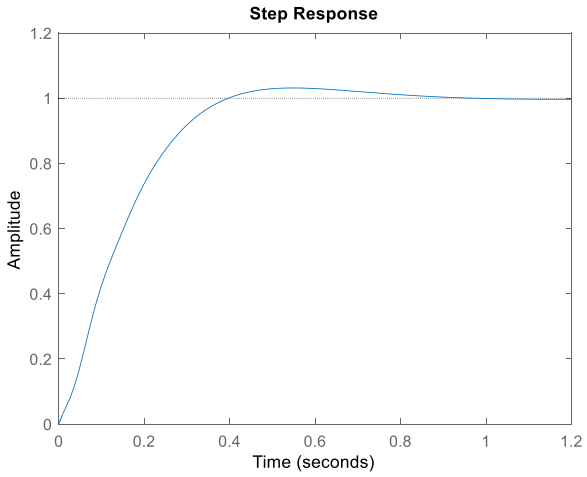
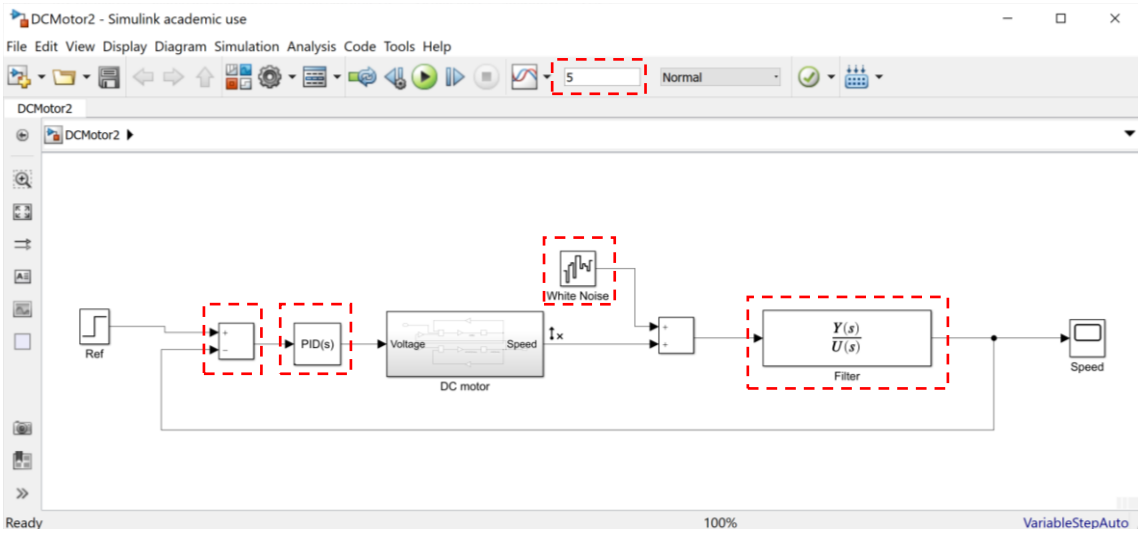


Figure 4.13: Step response of the Simulink model of the DC motor with Butterworth filter and PID controller.

Open the file `DCMotor2b.slx`, insert a block **PID Controller** from the library **Simulink \ Continuous** and connect its input to the line Voltage. Double click on the block and set the values Proportional to 100, Integral to 150, and Derivative to 10 on the Main tab.

Insert a block **Add** from the library **Simulink \ Math Operations**. Double click on it and enter the values "+-" in the **List of Signs**. This will create a positive input and a negative input. Connect the output of the Add block to the input of the PID block. Connect the inputs of the Add block to the step block and to the speed as shown in the figure below. Save the file as `DCMotor2d.slx`



Run the simulation and double click on the scope: the step response should be similar to the figure below. Note that, thanks to the filter, the effect of the measurement noise is hardly noticeable.

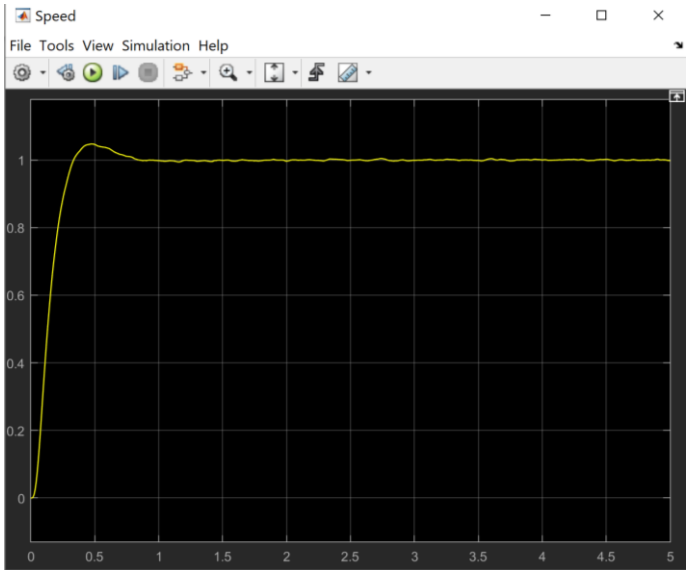


Figure 4.14: Step response of the Simulink model of the DC motor with the PID controller, white noise source, and Butterworth filter.

Questions:

For K_p = 100, K_i = 150, K_d = 10;

Q 24 Upload the closed loop step response (Figure 4.13) on **Blackboard**

Q 25 Does the overshoot satisfy design criteria (<10%)?

Q 26 Does the steady state error satisfy design criteria (<2%) ?

Q 27 Does the rise time satisfy design criteria (<1s)?

END OF VL-3
